

스터디 3주차 시간에는 4장 스택에 대해서 공부하고 공통문제로 교재의 13번과 16번을 풀어보았고 13,16번 이외에도 추가로 연습문제를 풀어보았습니다.
자료구조 추가문제 1주차 공동문제도 함께 보고서에 제출하겠습니다.

연습문제 10

배열에 들어있는 정수의 순서를 거꾸로 하는 프로그램을 스택을 이용해 작성하라.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#define MAX_STACK_SIZE 100

typedef int element;
typedef struct {
    element data[MAX_STACK_SIZE];
    int top;
} StackType;

// 스택 초기화 함수
void init_stack(StackType* s)
{
    s->top = -1;
}

// 공백 상태 검출 함수
int is_empty(StackType* s)
{
    return (s->top == -1);
}

// 포화 상태 검출 함수
int is_full(StackType* s, int *size)
{
    return (s->top == (*size - 1));
}
```

```

// 삽입함수
void push(StackType* s, element item, int *size)
{
    if (is_full(s,size)) {
        fprintf(stderr, "스택 포화 에러\n");
        return;
    }
    else s->data[++(s->top)] = item;
}

// 삭제함수
element pop(StackType* s)
{
    if (is_empty(s)) {
        fprintf(stderr, "스택 공백 에러\n");
        exit(1);
    }
    else return s->data[(s->top)--];
}

void get_n(StackType *s, int size) {
    printf("정수를 입력하시오: ");
    while (!is_full(s, &size)) {
        int n;
        scanf("%d", &n);
        push(s, n, &size);
    }
}

int main() {
    StackType s;
    init_stack(&s);

    printf("정수 배열의 크기: ");
    int size;
    scanf("%d", &size);

    get_n(&s, size);

    printf("반전된 정수 배열: ");
    while (!is_empty(&s)) {
        int pop_n= pop(&s);
        printf("%d ", pop_n);
    }
}

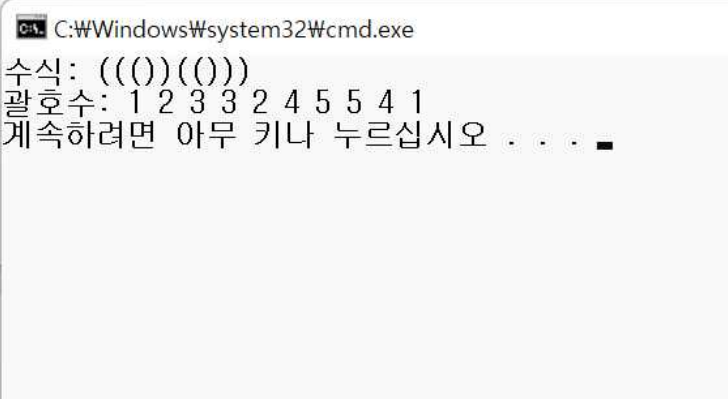
```

```
printf("\n");  
}
```

연습문제 11

수식에 있는 괄호의 번호를 출력하는 프로그램을 작성하라. 왼쪽괄호가 나올 때마다 괄호 번호는 하나씩 증가한다. 오른쪽 괄호가 나오면 매칭되는 왼쪽 괄호 번호를 출력한다.

<실행화면>



```
C:\Windows\system32\cmd.exe  
수식: (((()))(())  
괄호수: 1 2 3 3 2 4 5 5 4 1  
계속하려면 아무 키나 누르십시오 . . .
```

```
#define _CRT_SECURE_NO_WARNINGS  
  
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
  
#define MAX_STACK_SIZE 100  
  
typedef int element;  
typedef struct {  
    element data[MAX_STACK_SIZE];  
    int top;  
} StackType;  
  
// 스택 초기화 함수  
void init_stack(StackType* s)  
{  
    s->top = -1;  
}  
  
// 공백 상태 검출 함수  
int is_empty(StackType* s)  
{  
    return (s->top == -1);  
}  
  
// 포화 상태 검출 함수
```

```

int is_full(StackType* s)
{
    return (s->top == (MAX_STACK_SIZE - 1));
}
// 삽입함수
void push(StackType* s, element item)
{
    if (is_full(s)) {
        fprintf(stderr, "스택 포화 에러\n");
        return;
    }
    else s->data[++(s->top)] = item;
}
// 삭제함수
element pop(StackType* s)
{
    if (is_empty(s)) {
        fprintf(stderr, "스택 공백 에러\n");
        exit(1);
    }
    else return s->data[(s->top)--];
}
// 피크함수
element peek(StackType* s)
{
    if (is_empty(s)) {
        fprintf(stderr, "스택 공백 에러\n");
        exit(1);
    }
    else return s->data[s->top];
}

void check_match(char match[], int size) {
    StackType s;
    init_stack(&s);

    char c;
    int count = 0;
    for (int i = 0; i < size; i++) {
        c = match[i];
        switch (c) {
            case '(':
                count++;

```

```

        push(&s, count);
        printf("%d ", peek(&s));
        break;
    case ')':
        printf("%d ", pop(&s));
        break;
    }
}
printf("\n");
}

int main() {
    char match[MAX_STACK_SIZE];
    printf("수식: ");
    scanf("%s", match);

    int len = strlen(match);

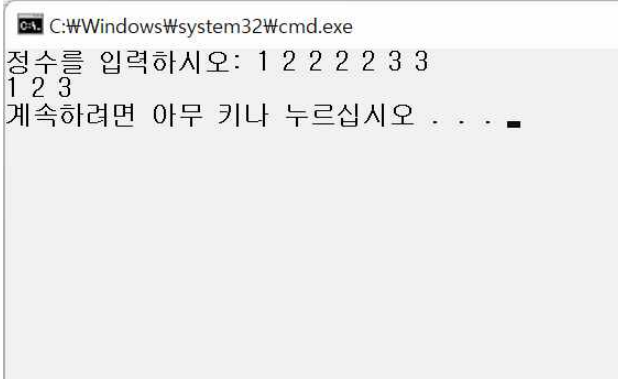
    printf("괄호수: ");
    check_match(match, len);
}

```

check_match()함수에서 스택을 초기화 하고 입력한 문자열의 길이만큼 for루프를 돈다. 문자가 만약 왼쪽 괄호라면 괄호의 번호를 지정하는 count 변수를 늘린다. 그 count변수(괄호의 번호)를 스택에 push한 뒤 가장 맨 위에 있는 요소를 peek하여 출력한다. 오른쪽 괄호라면 매칭되는 왼쪽 괄호의 번호를 출력해야 하니 스택에서 pop하여 출력한다.

연습문제 13

주어진 정수에서 반복되는 숫자를 제거하는 프로그램을 스택을 사용해 작성하라.



```

C:\Windows\system32\cmd.exe
정수를 입력하시오: 1 2 2 2 2 3
1 2 3
계속하려면 아무 키나 누르십시오 . . .

```

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

```

```

#include <stdlib.h>
#include <string.h>
#define MAX_STACK_SIZE 6

typedef int element;
typedef struct {
    element data[MAX_STACK_SIZE];
    int top;
} StackType;

// 스택 초기화 함수
void init_stack(StackType* s)
{
    s->top = -1;
}

// 공백 상태 검출 함수
int is_empty(StackType* s)
{
    return (s->top == -1);
}

// 포화 상태 검출 함수
int is_full(StackType* s)
{
    return (s->top == (MAX_STACK_SIZE - 1));
}

// 삽입함수
void push(StackType* s, element item)
{
    if (is_full(s)) {
        fprintf(stderr, "스택 포화 에러\n");
        return;
    }
    else s->data[++(s->top)] = item;
}

// 삭제함수
element pop(StackType* s)
{
    if (is_empty(s)) {
        fprintf(stderr, "스택 공백 에러\n");
        exit(1);
    }
    else return s->data[(s->top)--];
}

```

```

}
// 피크함수
element peek(StackType* s)
{
    if (is_empty(s)) {
        fprintf(stderr, "스택 공백 에러\n");
        exit(1);
    }
    else return s->data[s->top];
}

void delete_num(StackType s) {
    StackType t;
    init_stack(&t);

    push(&t, pop(&s));

    while (!is_empty(&s)) {
        if (peek(&t) != peek(&s))
            push(&t, pop(&s));
        else
            pop(&s);
    }

    while (!is_empty(&t))
        printf("%d ", pop(&t));
    printf("\n");
}

int main() {
    StackType s;
    init_stack(&s);
    printf("정수를 입력하시오: ");
    while (!is_full(&s)) {
        int n;
        scanf("%d", &n);
        push(&s, n);
    }
    delete_num(s);
}

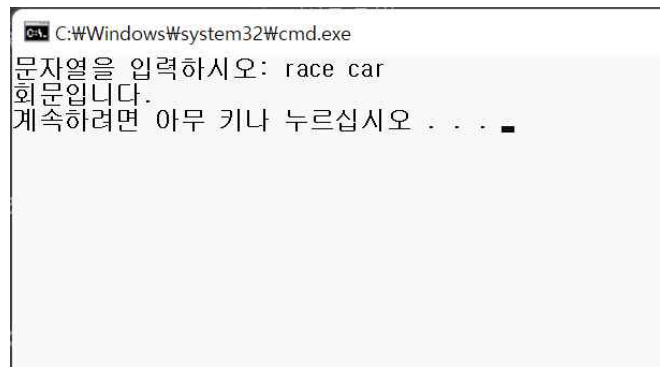
```

먼저 스택의 최대 사이즈를 6으로 가정하여 6개의 정수만 입력받도록 하였다. main에서 스택을 초기화하고 스택이 꽉 차지 않을 동안 정수를 입력받고 입력받은 정수를 스택에

push하였다. void타입의 delete_num()함수에서 StackType t를 선언하고 초기화하였다. 매 개변수로 받은 스택 s에서 먼저 pop하여 스택t에 push한다. 그런 다음 스택s가 공백 상태가 아닐 동안 peek을 이용하여 스택 s의 맨 위의 요소와 스택 t의 맨 위의 요소가 같지 않으면 중복된 숫자가 아니라는 뜻이니 스택s에서 pop하여 스택t로 push해준다. 그렇지 않다면 스택 s의 맨 위의 요소와 스택 t의 맨 위의 요소가 같다면 중복된 숫자가 있다는 뜻이니 스택 t에 push 하지 않고 스택 s에서 pop만 해주면 된다. 스택 s가 공백 상태가 되면 스택 t에는 중복된 숫자는 제거된 채 스택 s에서 입력된 숫자의 역순으로 저장되어있다. 따라서 스택 t가 공백이 아닐 동안 pop한 값을 출력해주면 1 2 3이 출력된다.

연습문제 16

회문(palindrome)이란 앞뒤 어느 쪽에서 읽어도 같은 단어를 의미한다. 예를 들면 “madam”, “race car”등이다. 여기서 구두점이나 스페이스 대소문자 등은 무시하여야 한다. 스택을 이용하여 주어진 문자열이 회문인지 아닌지를 결정하는 프로그램을 작성하라.



```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#define MAX_STACK_SIZE 100

typedef int element;
typedef struct {
    element data[MAX_STACK_SIZE];
    int top;
} StackType;

// 스택 초기화 함수
void init_stack(StackType* s)
{
    s->top = -1;
}
  
```



```

// 공백 상태 검출 함수
int is_empty(StackType* s)
{
    return (s->top == -1);
}
// 포화 상태 검출 함수
int is_full(StackType* s)
{
    return (s->top == (MAX_STACK_SIZE - 1));
}
// 삽입함수
void push(StackType* s, element item)
{
    if (is_full(s)) {
        fprintf(stderr, "스택 포화 에러\n");
        return;
    }
    else s->data[++(s->top)] = item;
}
// 삭제함수
element pop(StackType* s)
{
    if (is_empty(s)) {
        fprintf(stderr, "스택 공백 에러\n");
        exit(1);
    }
    else return s->data[(s->top)--];
}
// 피크함수
element peek(StackType* s)
{
    if (is_empty(s)) {
        fprintf(stderr, "스택 공백 에러\n");
        exit(1);
    }
    else return s->data[s->top];
}

int is_palindrome(char name[]) {
    StackType s;
    init_stack(&s);

```

```

int len = strlen(name);
int count = 0;
for (int i = 0; i < len; i++) {
    char ch = name[i];
    if (isupper(ch))
        ch = tolower(ch);
    if (ch != ' ' && ch != '.' && ch != ',')
        push(&s, ch);
    else
        count++;
}

StackType t;
init_stack(&t);

len -= count;
for (int i = 0; i < len / 2; i++) {
    push(&t, pop(&s));
}

if (len % 2 == 1)
    pop(&s);

while (!is_empty(&s) && !is_empty(&t)) {
    if (pop(&s) != pop(&t))
        return 0;
}
return 1;
}

int main() {

    char name[100];
    printf("문자열을 입력하시오: ");
    gets_s(name, 100);

    if (is_palindrome(name) != 1)
        printf("회문이 아닙니다.\n");
    else
        printf("회문입니다.\n");
}

```

is_palindrome()함수는 입력받은 문자열이 회문이면 1을, 회문이 아니면 0을 반환하

는 함수이다. 먼저 StackType s를 선언하고 스택을 초기화한다. 입력받은 문자열의 길이를 len으로 받아 for루프를 돌고 문자를 하나씩 받아 ch라는 변수에 저장한다. 만약 ch가 대문자라면 소문자로 바꿔주도록 inupper()함수와 tolower()함수를 이용하였다. 만약 ch가 공백 문자와 구두점이 아니라면 스택 s에 ch를 push하고 공백문자 또는 구두점이면 그 개수를 count로 세주었다. StackType t를 선언하고 초기화 해준뒤 구두점의 개수 count를 len에서 빼주었다. 회문은 앞뒤가 똑같으니 len의 절반만큼만 for루프를 돌고 스택 s에 저장된 문자를 pop하여 스택 t에 저장하면된다. 만약 len이 홀수라면 스택 s의 저장된 개수가 스택 t의 저장된 값보다 하나 많게 되니 s에서 하나 pop해주면 된다. 그리고 스택 s와 스택 t가 공백 상태가 아닐 동안 스택s와 스택t를 pop한 값을 비교하며 두 값이 같지 않으면 return 0를 해주어 함수에서 빠져나가면 된다. 두 스택이 공백상태가 될 때 까지 두 스택에서 pop한 값이 같으면 회문이라는 뜻이니 return 1을 해주면 된다.

<추가문제 climbStairs>

```
C:\Windows\system32\cmd.exe
5
6
climbStairs(6) = 13
5
climbStairs(5) = 8
4
climbStairs(4) = 5
3
climbStairs(3) = 3
2
climbStairs(2) = 2
계속하려면 아무 키나 누르십시오 . . .
```

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int climbStairs(int n) {
    if (n < 2)
        return 1;
    else
        return climbStairs(n - 1) + climbStairs(n - 2);
}

int main() {
    int T;
    scanf("%d", &T);
    while (T != 0) {
        int n;
        scanf("%d", &n);
```

```
        printf("climbStairs(%d) = %d\n", n, climbStairs(n));
        T--;
    }
}
```

먼저 main에서 테스트 케이스 T를 입력받고 테스트 케이스가 0이 될 때 까지 n을 입력받는다. 1번의 단계에서 계단을 한 칸 혹은 두 칸을 이동할 때 n번째의 계단을 가기 위해 필요한 단계를 리턴하는 climbStairs함수에서 만약 계단이 두 칸 미만일때는 1번의 단계 밖에 거치지 못하기 때문에 1을 리턴해주고. n이 2이 이상일때는 $\text{climbStairs}(n - 1) + \text{climbStairs}(n - 2)$ 을 리턴해준다. 만약 n이 2라면 climbStairs(1)이 먼저 호출되고 이 때의 n은 1임으로 2보다 작기 때문에 1이 리턴된다. 그 후 climbStairs(0)이 호출되어 마찬가지로 1이 리턴되며 climbStairs(1)와 climbStairs(0)을 더한 2가 climbStairs(2)에서 리턴된다. n이 3, 4, 5, 6일 때도 마찬가지로 같은 구조가 적용되기 때문에 문제의 구조는 같고 크기만 작아져 순환을 적용하여 문제를 해결 할 수 있다.